RT-Xen: Real-Time Virtualization in Xen

Presented by: Waqar Ali
EECS-768
Agenda

Main Sections

• Problem Statement
• Background
• RT-Xen
• Demonstration
• Active Research Topics
• Conclusion
Agenda

Main Sections

• **Problem Statement**
  - Contemporary Utility of System Virtual Machines
  - Need for Real-Time Performance

• Background

• RT-Xen

• Demonstration

• Active Research Topics

• Conclusion
Problem Statement

Contemporary Utility of System Virtual Machines

• Purpose of System Virtual Machines
  • Support multiple environments (users) on a single machine

Utility

• Hardware is becoming increasingly more capable
  • Example: Intel Xeon E7-8894 (48 Cores, 60 MB L3 Cache, Up-to 3-TB Main Memory)

• Single user not enough to employ all resources

• Solution: Virtualization!

• Use-Case: Amazon Cloud
  • Multiple users utilizing high-end physical (server) machines in virtualized environments
Problem Statement

Contemporary Utility of System Virtual Machines

• Similar advancements in COTS embedded platforms

  • **SWaP (Size, Weight and Power)** requirements are leading to **more consolidation**

  • **Example:** NVIDIA Jetson Xavier *(8-Cores, 4-MB LLC, 512-Core Volta GPU, 2 DL-Accelerators, VLIW Vision Processor, 16-GB Main Memory)*

• **Use-Case:** *Virtualization for Cars*

  • 100s of ECUs ➔ N-physical processors

  • Integrate multiple systems on a common platform (e.g., Xavier)
Problem Statement

The Need for **Real-Time** Performance

- Different systems have different QoS requirements
  - Infotainment System - Best-Effort Requirement
  - Driver Assistance System - Soft Real-Time Requirement
  - Emergency Braking System - Hard Real-Time Requirement

Reference: [https://www.edn.com/design/automotive/4399434/ Multicore-and-virtualization-in-automotive-environments](https://www.edn.com/design/automotive/4399434/)
Problem Statement

The Need for **Real-Time** Performance

“In the virtualized environment, the **real-time requirements** of the *individual systems being virtualized* must be respected.”

- **Example**
  - **VM-1**: Braking system requires **5-ms response latency**
  - **VM-2**: Lane keeping assist requires **bounded tardiness** (*the extent to which a job can miss its deadline*)
  - **VM-3**: Streaming application requires a 20-FPS **throughput**
  - Xen, by default, cannot guarantee this under the credit based scheduling paradigm
    - Each VM will get 1/3 of the CPU credit (**Proportional Fairness**)
    - Hard real-time tasks cannot be **analytically** guaranteed their deadlines
Agenda

Main Sections

• Problem Statement

• **Background**
  
  • **Meaning of Real-Time**
  
  • **Concepts from Scheduling Theory**
    
    • **Fair Scheduling**
    
    • **Real-Time Scheduling**
  
  • RT-Xen

• Demonstration

• Active Research Topics

• Conclusion
Background

Meaning of Real-Time

• Traditional meaning of Correctness
  
  • Given an input, the system computes the logically right response
  
  • Example: 2+2=4

• Real-Time Correctness
  
  • Given an input, the system computes the logically right response within a deterministic amount of time
  
  • Example: An autonomous car applies brakes within 30-usec of detecting an object in its way

• A logically correct output at a wrong time is a fault!
• **Purpose of Scheduling:** Sharing of a resource among multiple clients

• **Example:** CPU scheduling shares the CPU (resource) among multiple processes (clients)

• Different scheduling schemes can provide different guarantees

• **Fairness:** Given N clients, each receives $\frac{1}{N}$th portion of the resource

• **Real-Time Response:** Given N clients, the response time constraints of each can be satisfied based on their priority
Background

**Fair Scheduling**

- **Purpose:** Ensure proportional fairness among clients

- **Example:** CFS (Completely Fair Scheduler) in Linux, Credit Scheduler in Xen

**Fair Scheduling Illustration**

- Two VCPUs on a single-core system

- Scheduling Granularity: 6-msec

- At each 6-msec boundary, both VCPUs would have been given *equal amount of CPU time*
Background

Real-Time Scheduling

• **Purpose:** Each client can execute its job with **deterministic** latency

• **Example:** FIFO, Round-Robin, Deadline in Linux

• **RT Scheduling (FIFO) Illustration**

  • Two VCPUs on a single-core system, VCPU-1 high priority, VCPU-2 low priority

  • VCPU-1 guaranteed execution on physical core whenever ready

  • Classic response time analysis (RTA) can be applied to **analytically verify** schedulability
“In a virtualized environment, real-time schedulability of the system cannot be guaranteed unless the scheduler in each layer of abstraction is using a real-time policy.”

- Xen, with credit scheduler, can provide no guarantee on the latency of the jobs running in each VM.

- **Illustration**
  - System virtualized by Xen (Credit Scheduler)
  - VM-1 running a real-time OS with FIFO policy
  - VM-2 running Linux with CFS policy
  - Tasksets in VM-1 cannot be guaranteed real-time performance since the root scheduler is not real-time
Agenda

Main Sections

• Problem Statement
• Background

• RT-Xen
  • Features
  • Bringing RT to Xen
  • Scheduling Policies
    • Development over the Years
• Demonstration
• Active Research Topics
• Conclusion
RT-Xen

Features

• Real-Time VMM based on Xen
  • Real-Time CPU Sharing among VMs
  • Real-Time arbitration of IO / Network resources
• Spatial Isolation among VMs at Hardware Level
  • Cache Level Isolation
  • Memory Level Isolation
• Built on Compositional Scheduling Paradigm
  • Provides real-time guarantees to tasks inside individual VMs
• Open-Source
RT-Xen

Bringing RT to Xen

- **Xen**: Baremetal (i.e., Classic) System Virtual Machine
- Runs *paravirtualized* or *fully virtualized* OSes

- **VMM Scheduler**: Credit based
  - Assign Round-Robin quanta of execution on physical CPUs to each client (i.e., VCPU) to ensure proportional fairness

RT-Xen

• RT-Xen
  • Each virtual machine is considered a real-time resource interface defined by parameters
    • Period, Budget, # of VCPUs
  • The runtime parameters assigned to each interface are determined based on its requirement using Composition Scheduling Theory
  • Each VM is scheduled with its assigned parameters using a real-time scheduling policy
RT-Xen

Scheduling Policies

- Supports both **Global** and **Partitioned** scheduling
  - Dictates placement of VCPUs
- Allows selection of **Static** and **Dynamic** scheduling schemes
  - Dictates ordering of VCPUs
- **Static**: RMS (FIFO, RR etc.)
- **Dynamic**: EDF
- Allows **Server** based scheduling schemes
  - **Mechanism for Resource Isolation**
  - **Periodic, Polling, Deferrable, Sporadic**
RT-Xen

Scheduling Policies: Development over the Years

• RT-Xen 1.0
  • Introduced Single-Core RT scheduling

• RT-Xen 2.0
  • Multi-Core RT scheduling
    • RT-global
    • RT-partition
Agenda

Main Sections

- Problem Statement
- Background
- RT-Xen
- **Demonstration**
  - Xen vs RT-Xen
- Active Research Topics
- Conclusion
Demonstration

Xen vs RT-Xen

• **Setup**
  
  • **Platform:** Intel i7 Processor, 6 cores, 3.3 GHz
  
  • **Software:** Xen-4.3 patched with RT-Xen
  
  • **Workload:** Periodic real-time tasksets with increasing utilization
  
  • **Metric:** Proportion of tasksets which are **schedulable**
    
    • If all tasksets (i.e., 100%) of a given utilization are schedulable, the algorithm is said to satisfy **real-time schedulability requirement**
Demonstration

Xen vs RT-Xen

- Credit misses deadlines at 22% of CPU capacity.
- RT-Xen delivers real-time performance at 78% of CPU capacity.

Agenda

Main Sections

- Problem Statement
- Background
- RT-Xen
- Demonstration
- **Active Research Topics**
  - Hardware Level Resource Isolation
- Conclusion
Active Research Topics

• In Multicore Platforms, resource sharing in hardware can lead to unpredictable behavior

• Example

  • Dual core system with shared Last-Level Cache (LLC)
  • Miss in LLC incurs 100x penalty to memory request
  • Application running in VCPU of Core-1 fits in LLC
  • Application running in VCPU of Core-2 thrashes the LLC
  • When run simultaneously, the execution time of application on Core-1 will increase 100x

  • Real-Time guarantees cannot be satisfied in an unmanaged system
Active Research Topics

Hardware Level Resource Isolation

“In VMM context, real-time scheduling of CPUs is not enough to provide guaranteed latencies to virtual guests.”

- Ensure isolation among VMs at hardware level
  - **Common Solution:** Partitioning

- Partitioning in RT-Xen
    - Mechanism to partition shared LLC among virtual guests in RT-Xen in Intel Haswell processors
    - Memory level isolation among VMs using page-coloring
Conclusion

Key Takeaways

- Xen provides a mechanism to effectively utilize immensely capable emerging hardware platforms among multiple users

- RT-Xen ensures real-time schedulability of virtual guests in Xen

  - Implements real-time scheduling policies
  
  - Ensures hardware level isolation among virtual guests
  
  - Immensely improves real-time schedulability over the default credit scheduler of Xen
QUESTIONS?